ND Path Finder Executive Summary

Team Members: Joseph Schmidt, Alborz Ranjbar, Aoran Wu, Dilan Karaguler, Yangxiao Luo, Jingheng Wang
Github: https://github.com/joeyschmidt97/ND_north_star

Goal:
Develop an algorithm capable of detecting boundaries and finding pathways in 2D spaces using minimal data.

Boundary Detection

- Create a machine learning algorithm that detects the edges of data points, distinguishing between black regions and white regions using a minimal number of data points.

Major Priorities:

- **Performance Evaluation**: Establish a methodology to measure the accuracy and efficiency of the boundary detection and pathfinding algorithm.
- **Efficiency**: Ensure that the system processes data and retrieves results rapidly, ideally in sub-second time.

Data:
We generated data using a 2D Perlin noise algorithm on a 30x30 grid with 4 octaves for added detail and complexity. Perlin noise creates smooth, natural transitions. Using 4 octaves introduces multiple noise layers, each adding finer details and making a more complex pattern.

This approach ensures the data has a realistic, detailed structure that strives to mimic the complexity of physical systems (i.e. phase diagrams showing states of water at different temperatures and pressures). We generate a full Perlin noise 2D image then drop a percentage of the data reflecting the often timely/costly nature of collecting data at various parameters. Since the full 2D Perlin image is crucial for generating accurate edge boundaries for model benchmarking, we take great care to prevent any data leakage that could compromise model performance.

Evaluation Methods:
We use Weighted Cross Entropy (WSE) to evaluate the structure of the predicted boundary (i.e. width and continuity) compared to the actual boundary while the Mean Square Error (MSE) is used to holistically evaluate the average error of the image (both white and black dot placement).

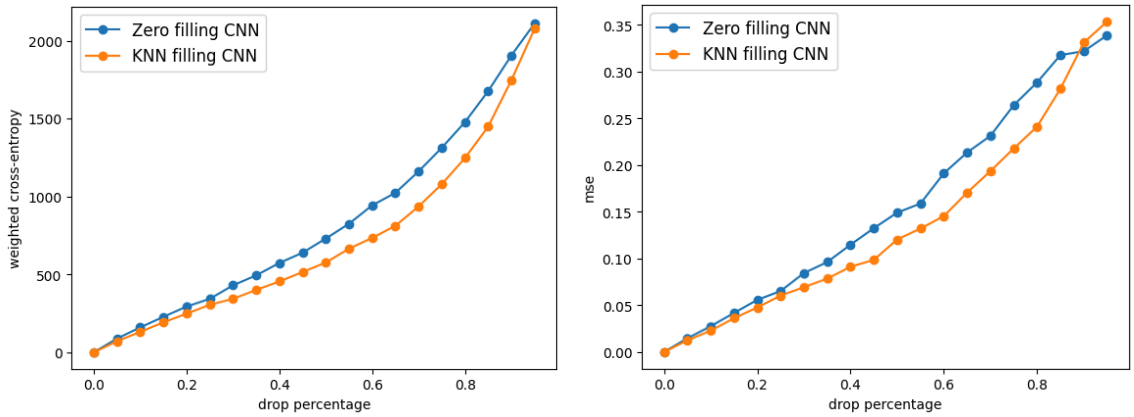
Models:

- **Zero-Filling-CNN model:**
  We fill any missing data with 0s to complete the image in order to feed it into our convolutional neural network (CNN) model. Two convolution layers, one MaxPooling layer, then two deconvolution layers. We use the Sigmoid function as our activation

function in the last layer to get values between 0 and 1. A deconvolutional layer is the output layer ensuring an image with the boundaries is given as the output

- **KNN-CNN model:**
  We use k-nearest neighbor (kNN), with the 3 nearest neighbors, to fill the missing data points and use the same CNN model as above to reconstruct the boundary given minimal information

Comparison of Models:



WSE (Left) and MSE (Right) comparison between zero in-fill CNN and kNN CNN

- WCE diverges while the MSE converges: Although we lose some features in the whole picture, we still guarantee pointwise accuracy at a certain level
- Zero-filling outperforms at the top of MSE: In extreme case, zero-filling guarantee 50% accuracy when kNN is out of the threshold of good performance

Future Directions:

- **Higher Dimensional Generalization:**
  - Assessing whether our model can generalize to higher dimensions and identifying suitable alternative models if it cannot. This will help us understand the scalability of our approach.
- **Pathfinding**
  - **Path Volume**: Determine the "volume" of a path by employing hyperspheres to measure the available space between boundaries.
  - **Path Windiness**: Assess the windiness of the path by analyzing vector angles in multiple dimensions as the path progresses.
- **Different Measurements:**
  - Hausdorff Distance (extreme geometric error between two sets of points)
  - Chamfer Distance (average geometric error between two sets of points)